IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 6, 2021, accepted December 21, 2021, date of publication December 23, 2021,
date of current version January 5, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3138130

Hybrid PSO-αBB Global Optimisation for C^2 Box-Constrained Multimodal NLPs

YVES MATANGA[1], YANXIA SUN[1], (Member, IEEE), AND ZENGHUI WANG[2], (Member, IEEE)
[1]Department of Electrical and Electronic Engineering Science, University of Johannesburg, Johannesburg 2006, South Africa
[2]Department of Electrical and Mining Engineering, University of South Africa, Johannesburg 1710, South Africa

Corresponding author: Zenghui Wang (wangz@unisa.ac.za)

ABSTRACT αBB is an elegant deterministic branch and bound global optimisation that guarantees global optimum convergence with minimal parameter tuning. However, the method suffers from a slow convergence speed calling for computational improvements in several areas. The current paper proposes hybridising the branch and bound process with particle swarm optimisation to improve its global convergence speed when solving twice differentiable (C^2) box-constrained multimodal functions. This hybridisation complemented with interval analysis leads to an early discovery of the global optimum, quicker pruning of suboptimal regions in the problem space, thus improving global convergence. Also, when used as a heuristic search algorithm, the hybrid algorithm yields superior solution accuracy owing to the combined search capabilities of PSO and the branch and bound framework. Computational experiments have been conducted on CEC 2017/2019 test sets and on n-dimensional classical test sets yielding improved convergence speed in the complete search configuration and superior solution accuracy in the heuristic search configuration.

INDEX TERMS αBB, particle swarm optimisation (PSO), box-constrained NLPs, hybridisation.

I. INTRODUCTION

Deterministic global optimisation techniques are rigorous and complete search algorithms [1] that aims to find an $\epsilon-$accurate global optimum solution in finite time unlike its stochastic counterpart (particle swarm optimisation [2], genetic algorithm [3], differential evolution [4], simulated annealing [5], ant colony optimisation [6] and other competing techniques [7]). These techniques generally proceed by lower and upper bounding of the solution space using branch and bound frameworks [8]. Among existing deterministic global optimisation approaches, αBB is an elegant branch and bound (BB) framework [9], [10] that guarantees convergence to the true optimum for twice differentiable problems. However, the framework generally suffers from a slow convergence speed owing to its often loose convex relaxation. Several research works have been done in the literature [11]–[15] aiming to improve bound tightness and thus boost the overall computational efficiency. In the same tune, interval analysis is also used as a zero-order convex relaxation that supplements α-convex relaxation and provide tighter lower bounds [16]. While several research works have been centred on bound tightness, not so much research work has been performed in other areas of the procedure, including the performance of the upper bound solver. In a bid to further improve convergence speed, the current study looks at the exploration capability of the upper bound solver that typically makes use of local search algorithms. The study proposes an efficient use of particle swarm optimisation (PSO) as an upper bound solver in the BB-procedure to allow early discovery of the true optimum, which could lead to early pruning of suboptimal regions and speed up global convergence provided the availability of tight bounds supplemented by interval analysis. In addition, the hybrid algorithm can be used as a heuristic search algorithm combining the search capability of PSO with that of the branch and bound framework to which a deterministic stopping criterion is added, thus yielding a solver that is beneficial to both frameworks, stochastic and deterministic. The main contributions of the paper are as follows:

1) A novel PSO-αBB hybrid algorithm is proposed that improves the convergence speed of classical αBB for C^2 multimodal bound-constrained problems.

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

2) A PSO via branch and bound framework heuristic search algorithm is proposed with improved solution accuracy.
3) Additional insights on the bound tightness performance of α-convex relaxation and classical interval analysis are provided and discussed.

The rest of this paper is organised as follows: Section 2 elaborates on the αBB search procedure presenting the configuration of the state of the art inclusive of all pruning rules. Section 3 describes the particle swarm optimisation algorithm, including critical components related to this study. Section 4 presents the proposed hybrid algorithm. Section 5 describes the computational experiments used to assess the performance of the proposed algorithm, discusses the findings and proposes future research areas. Section 6 provides a conclusion to the study.

II. DETERMINISTIC GLOBAL OPTIMISATION: αBB

A. CLASSICAL ALGORITHM

Deterministic global optimisation typically proceeds by an exhaustive partitioning of the solution space in which upper and lower bounding of the problem inner regions anticipates pruning of sub-optimal areas in the search for the $\epsilon-$global optimum. This divide and conquer approach is performed within a branch and bound framework [8]. αBB is a branch and bound framework that offers an elegant lower bounding scheme that creates an underestimator by adding a function term to the initial problem with minimal inclusion of additional parameters (α) applicable to twice differentiable functions. The algorithm was developed by [17], [18]. A convex underestimator f^l to a nonlinear function f is obtained by adding a quadratic function to the original non-convex function in such a way that it overpowers any non-convexity in the original function:

$$f^l = f(\mathbf{x}) - \sum_{i=1}^{N} \alpha_i (x_i - {x_i}^l)(x_i - {x_i}^u) \tag{1}$$

where x_i^l and x_i^u are the bounds on the dimensions of the solution space and N, the dimension order of the problem. The formulation of the convex understimator relies on the accurate estimation of the α vector, which requires that the Hessian matrix H_{f^l} be semi-definite:

$$H_{f^l}(x) = H_f(x) + 2\Delta \tag{2}$$

where Δ is a diagonal matrix whose diagonal elements are elements of the α vector and H_f, the hessian matrix of the original problem. Theoretically, elements of the α vector can be obtained by finding the smallest eigenvalues in each dimension of the hessian matrix of f within the bounded region [18]:

$$\alpha_i = \max\{0, -\frac{1}{2}\min_{i,x^l \leq x \leq x^u} \lambda_i(x)\} \tag{3}$$

where $\lambda_i(x)$'s are the eigenvalues of $H_f(x)$ in the given interval. One efficient way to solve equation 3 is by using interval analysis that finds the interval Hessian matrix $[H_f] \supseteq H_f(x)$ such that

$$\alpha_i = \max\{0, -\frac{1}{2}\min_{i,x^L \leq x \leq x^U} \lambda_i([H_f])\} \tag{4}$$

Among existing computation methods to find the α vector based on equation 4, the technique based on the scaled Gershgorin theorem [9], [10] is typically used such that the minimum eigenvalue of an interval matrix $[A] = ([a_{ij}^l, a_{ij}^u])$ is given by

$$\underline{\lambda_i} = \min_i[\underline{a_{ii}} - \sum_{j \neq i} \max(|\underline{a_{ij}}|, |\overline{a_{ij}}|)] \tag{5}$$

and thus

$$\alpha_i = max(0, -\frac{1}{2}\underline{\lambda_i}) \tag{6}$$

To alleviate the conservative nature of α-convex relaxation, it is supplemented with interval analysis [16] (i.e. $\text{LB}_i = \max(lb_i, \underline{f}))$. Interval analysis is a mathematical arithmetic that aims to provide a lower and upper bound to the range of a function given the domain of its input variables, that is, given $[\underline{x}, \overline{x}]$, find $[\underline{y}, \overline{y}]$ such that $f([\underline{x}, \overline{x}]) \subseteq [\underline{y}, \overline{y}]$. The methodology proceeds by defining interval arithmetic for elementary operators from which more complex expressions are built:

$$[\underline{x_1}, \overline{x_1}] + [\underline{x_2}, \overline{x_2}] = [\underline{x_1} + \underline{x_2}, \overline{x_1} + \overline{x_2}] \tag{7}$$

$$[\underline{x_1}, \overline{x_1}] - [\underline{x_2}, \overline{x_2}] = [\underline{x_1} - \overline{x_2}, \overline{x_1} - \underline{x_2}] \tag{8}$$

$$\begin{aligned}[\underline{x_1}, \overline{x_1}] * [\underline{x_2}, \overline{x_2}] = [&min(\underline{x_1}.\underline{x_2}, \underline{x_1}\overline{x_2}, \overline{x_1}\underline{x_2}, \overline{x_1}.\overline{x_2}),\\ &max(\underline{x_1}.\underline{x_2}, \underline{x_1}\overline{x_2}, \overline{x_1}\underline{x_2}, \overline{x_1}.\overline{x_2})]\end{aligned} \tag{9}$$

$$[\underline{x_1}, \overline{x_1}] \div [\underline{x_2}, \overline{x_2}] = [\underline{x_1}, \overline{x_1}].\frac{1}{[\underline{x_2}, \overline{x_2}]} \tag{10}$$

$$\frac{1}{[\underline{x_2}, \overline{x_2}]} = \begin{cases} \text{NaN} & \text{if } \underline{x_2} = \overline{x_2} = 0 \\ [\frac{1}{\overline{x_2}}, \frac{1}{\underline{x_2}}] & \text{if } 0 \notin [\underline{x_2}, \overline{x_2}] \\ [\frac{1}{\overline{x_2}}, +\infty[& \text{if } (\underline{x_2} = 0) \text{ and } (\overline{x_2} > 0) \\ ]-\infty, \frac{1}{\underline{x_2}}] & \text{if } (\underline{x_2} < 0) \text{ and } (\overline{x_2} = 0) \\ ]-\infty, +\infty[& \text{if } (\underline{x_2} < 0) \text{ and } (\overline{x_2} > 0) \end{cases} \tag{11}$$

Further details on more complex functions (cos, sin, asin, acos, log, etc.) can be found in [19]. While interval analysis also leads to overestimation in some instances due to the dependency problem [20], our computational experiments suggest that it often leads to much tighter bounds than α-convex relaxation and thus can validly complement the lower bounding process.

Convergence speed of the branch and bound process is accelerated by domain reduction techniques [21] and interval analysis based pruning rules [22], [23]. While domain reduction techniques are used selectively depending on the problem being solved, interval analysis based pruning rules can routinely be used to supplement the classical fathoming mechanism (See section II-B). Algorithm 1 describes a pseudo-code of a typical αBB procedure.

The minimum and maximum iteration limit to complete the branch and bound framework search is a function of the problem dimension size (n), the size of the solution space, the tightness of the underestimator (α), and the tolerance level required (ϵ) [18]:

$$I^{min} = n\log_2\left(\frac{\sum_{i=1}^{n}(x_i^u - x_i^l)^2}{\sqrt{\frac{4\epsilon}{\alpha}}}\right) - 1 \tag{12}$$

$$I^{max} = \left(\frac{\sum_{i=1}^{n}(x_i^u - x_i^l)^2}{\sqrt{\frac{4\epsilon}{\alpha}}}\right)^n - 1 \tag{13}$$

exhibiting linearithmic ($O(nlog(n))$) and exponential ($O(n^n)$) time and space complexity in the best and worst-case scenario respectively.

Algorithm 1: A Typical α BB-Procedure

```
Let $X_0 = [x^l, x^u]$, $\{X\} \leftarrow X_0$, BUB $= +\infty$, set $\epsilon$;
while $\{X\} \neq \emptyset$ do
    Select a node $X_j \in \{X\}$;
    Calculate $f_j = inf(f(x)), x \in X_j$;
    if $\underline{f_j} > BU\overline{B}$ then
        Fathom node $X_j$ and move to next iteration;
    end
    Calculate $[\nabla f]$ and $[H_f]$, $x \in X_j$;
    if $f$ is monotonous or non-convex then
        Reduce $X_j$ and move to next iteration;
    end
    Solve for $ub_j$ = local_solver(f(x)),$x \in X_j$;
    Update ($x^*$,BUB) = min($ub_j$,BUB);
    Compute $\alpha$ and set $f^l(x) = f(x) + CT(\alpha, x)$;
    Solve for $lb_j$ = local_solver($f^l(x)$),$x \in X_j$;
    if $lb_j > BUB$ then
        Fathom node $X_j$ and move to next iteration;
    end
    Update node lower bound: LB$_j$ =max($\underline{f_j}$, $lb_j$);
    if $ub_j - LB_j > \epsilon$ then
        $[X_{j1}, X_{j2}]$ = branch($X_j$), $\{X\} \leftarrow X_{j1}$, $\{X\} \leftarrow X_{j2}$
    end
    Fathom node $X_j$;
end
return ($x^*$, BUB);
```

B. PRUNING RULES

1) BOUND TEST

"If the lower bound of f in a given region V is above any upper bound outside region V, the region V does not contain the global optimum and can therefore be pruned".

2) MONOTONICITY TEST

"If f is strictly monotone in one of the variables over a box $V = V_1x\ldots xV_m$, $V \subset X$, the box V does not contain the global optimum of f over $X = X_1x\ldots xX_m$ except that the edge f in X touches the edge of V. Monotonicity can be tested by the sign of the interval gradient matrix of f in V":

$$\begin{aligned} &\text{If } \exists x_i,\ \text{a dim of } x : \forall x_i \in V_i, \nabla f(x_i) < 0 \\ &\implies f \text{ is strictly monotone decreasing} \\ &\text{If } \exists x_i,\ \text{a dim of } x : \forall x_i \in V_i, \nabla f(x_i) > 0 \\ &\implies f \text{ is strictly monotone increasing} \end{aligned}$$

The box V should be reduced to an edge piece on the monotonous dimension [23] and does not contain the global optimum except possibly on its edge piece.

3) NON-CONVEXITY TEST

"If f is concave over a given box $V \subset X$, the box V does not contain the global optimum except that the edges f in X touches the edges of V. Concavity can be tested by the sign of the diagonal elements of the interval Hessian matrix of f in V":

$$\text{If } \exists x_i,\ \text{a dim of } x : \forall x_i \in V_i, \nabla^2 f(x_i) < 0 \implies f \text{ is concave over } V$$

The box V can be pruned or reduced to its left or right edge piece on the concave dimension [23].

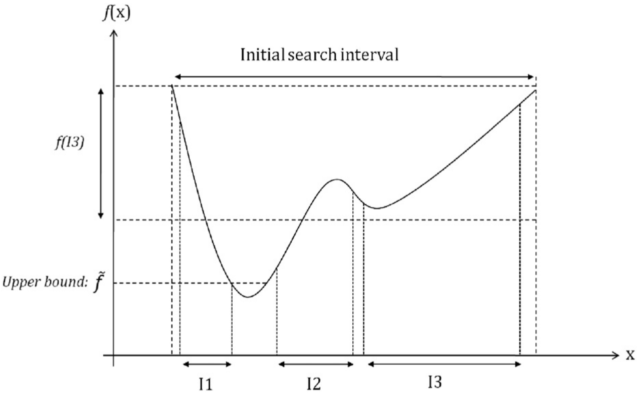


FIGURE 1. Pruning rules in box-constrained global optimisation. The following representation has been extracted from [22]. I_1 relates to the monotonicity test, and I_2 refers to the concavity test. I_3 relates to the bound test.

III. PARTICLE SWARM OPTIMISATION

Particle swarm optimisation is a population-based optimisation algorithm inspired by the foraging of flocks of birds in a collaborative methodology. PSO employs an interactive random search for the global optimum based on swarm intelligence. A set of randomly generated potential solutions is generated, and each individual (particle) iteratively improves its position based on its own experience (cognitive learning) and based on the experience of other individuals in the swarm (social learning). The motion of each particle in the search space is thus dictated by a collaborative stochastic search direction:

$$v_j^{k+1} = \omega^k v_j^k + c_1 r_1^k(p_j^k - x_j^k) + c_2 r_2^k(g_{(j)}^k - x_j^k) \tag{14}$$

$$x_j^{k+1} = x_j^k + v_j^{k+1} \tag{15}$$

where v_j^{k+1} is the search direction of a given particle for the next iteration, which is a function of its current velocity v_j^k, its best location thus far p_j^k (cognitive learning) and the best location $g_{(j)}^k$ (social learning) in the neighbourhood of the particle (g_j^k) or within the whole swarm (g^k). When $g_{(j)}^k$ represent the best particle location in a given neighbourhood, the algorithm is referred to as local, and it is referred to as global when $g_{(j)}^k$ represent the best candidate in the whole population [24]. The latter version is used in the study as our computational experiments suggested better time efficiency in line with the focus of the study. The parameters c_1 and c_2 represent acceleration parameters for the cognitive and social learning components. $r_1^k, r_2^k \in [0, 1]$ are uniform randomly generated numbers that simulate the stochastic behaviour of the swarm. Figure 2 illustrates geometrically how the motion of each particle within a swarm is dictated.

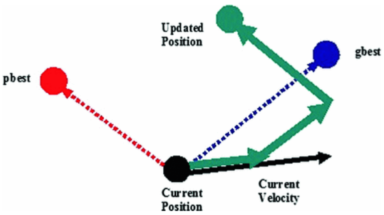


FIGURE 2. Geometric illustration of particle movement in PSO [25].

A. PARAMETER TUNING

Convergence speed and exploration are primarily dictated by the cognitive and social parameters c_1, c_2 as well as by the inertial weight ω^k. c_1 and c_2 decide the speed of each particle as well as the algorithm bias towards either exploitation (fast convergence) or exploration. If $c_1 = c_2 > 0$, the algorithm is balanced. The particles attract towards the average of p_j^k and g^k. When $c_1 > c_2$, the algorithm gives preference to exploration than quick convergence, whereas when $c_1 < c_2$, the algorithm favours exploitation over exploration. Smaller values of c_1 and c_2 lead to smooth particle trajectories during the search, while larger values lead to abrupt motion with more acceleration [26]. The original publication proposes the value of $c_1 = c_2 = 2$ and these settings are typically used in the literature [27]. The inertial parameter w^k defines how willing a given particle is in maintaining its current direction. It contributes to dictating bias towards exploration and exploitation as well. The higher the inertial parameter, the more exploratory the search. In the original publication, the inertial parameter was set to 1 ($w^k = w = 1$). However, a more adaptive variation w^k is accepted in recent literature [26], [28] that favours a high inertial weight value at the beginning of the search, which progressively drops over iterations:

$$w^k = w_{max} - \frac{w_{max} - w_{min}}{max\_iter} k \tag{16}$$

This methodology favours exploration at the beginning of the search and exploitation at the end of the search to eventually narrow the search down to the area containing the best fitness and explore it in detail. The value of $w_{max} = 0.9$ and $w_{min} = 0.4$ are typically used in the literature. Apart from the aforementioned parameters, three additional parameters must be set. The population size of the algorithm is also determinant to its exploration capability. A population size of 20-50 particles is routinely used in the literature and deemed satisfactory. The velocity bound of the particles are often determined as lying in a specified range $[-v_{max}, v_{max}]$ which limits the velocity of the particles. These velocity bounds are typically of the type [27], [29], [30]:

$$v_{imax} = \frac{x_i^u - x_i^l}{I_i} \tag{17}$$

where I_i is the scaling factor of the maximum velocity in the i^{th} dimension selected by the user. I_i has been set to 2 in the current study with a smooth swarm behaviour. Also, the position of the particle (x_j^{k+1}) is typically bounded not to exceed the problem dimension bounds.

B. STOPPING CRITERIA

Besides the use of a maximum iteration count, several heuristics have been proposed in the literature that could lead to an early stop of the algorithm when some features of the particles do not lead to reasonable improvement [30] which could save computation time. Improvement-based stopping criteria terminate the search when the improvement of the objective function (i.e. $f(g^k)$ or $\mu(f(p_j^k))$) is not significant for a number of iterations. Movement-based stopping criteria monitor the movement of particles for a given number of iterations. If the particle positions do not sensibly vary for a number of iterations, the search is halted. A gbest-based improvement criterion is used in this work as it is computationally efficient and is a satisfactory heuristic stopping criterion consistent with the focus of the study. A detailed survey on the topic can be found in [30]. Algorithm 2 presents the typical pseudo-code of the PSO search mechanism.

IV. HYBRIDISATION

In the view of the current hybridisation, PSO is used as an intelligent scatter search that favours an early discovery of the global optimum owing to its exploration capabilities, which could lead to early pruning of substandard regions in the BB-procedure, fewer nodes exploration and thus quicker convergence. It is well established in the literature that a best-first search strategy (A^* algorithm) in branch and bound procedures leads to fewer node exploration and fast convergence [8], [31]. The rationale of this idea is that a branch and bound procedure should first focus on regions estimated to contain the global optimum because, given tighter bounds, sub-optimal regions will subsequently be pruned and not explored. This philosophy is often the rationale that influences the common use of the best-first node selection strategy in BB-frameworks compared to other node selection approaches [8]. However, it does not

Algorithm 2: Typical PSO Procedure

```
Let $X_0 = [x^l, x^u]$, $V = [-v_{max}, v_{max}]$;
Set swarm size N, max_iter $K$ and $k = 0$;
Randomly generate initial population: $x_j \in X_0$;
Randomly generate initial velocities: $v_j \in V$;
Set $p_j^k$ for every particle to $x_j^k$;
Compute the swarm initial best point ($g^k$,BUB);
while $k < K$ and heuristic stop not reached do
    for $j$=1:N do
        $v_j^{k+1} = \omega^k v_j^k + r_1^k c_1(p_j^k - x_j^k) + r_2^k c_2(g^k - x_j^k)$;
        $v_j^{k+1} = \text{bound}(v^{k+1}, -v_{max}, v_{max})$;
        $x_j^{k+1} = x_j^k + v_j^{k+1}$;
        $x_j^{k+1} = \text{bound}(x^{k+1}, x^l, x^u)$;
        $f_j^{k+1} = f(x_j^{k+1})$;
        $p_j^k = \arg\min(\{f_j^{k+1}, f_j^k\})$;
        $g^k = \arg\min(\{f(p_j^k), \text{BUB}\})$;
    end
    $k = k + 1$;
end
return ($x^* = g^k$, BUB);
```

suffice to estimate the region that most likely contains the global optimum; the use of a local search in such a region could still lead to additional branching if such a solver is trapped in a local optimum (i.e. poor upper bound) while the lower bounding utility is rather tight. A more explorative search for the true optimum would increase the likelihood of exploring fewer nodes and eventually speed up convergence of the BB-procedure. Moreover, the use of PSO as an upper bound solver in the BB-procedure increases the likelihood that upon early termination of the branch and bound process due to a maximum iteration limit, a much-improved solution than the classical methodology can be obtained. From a PSO point of view, this hybridisation could guide the PSO solution search towards the true optimum owing to the αBB region partitioning scheme and node selection strategy, as well as provide a deterministic stopping criterion that informs the algorithm that the true optimum has been obtained, a feature that PSO alone cannot guarantee. Thus, the hybrid algorithm can also be used as a heuristic search algorithm that uses the combined search capabilities of PSO and αBB to obtain an improved solution. Several PSO-BB hybridisation approaches have been proposed in the literature yielding improved optimisation performances. [32] proposes PSO-BB hybridisation for solving integer separable concave programming problems using PSO as an upper bound solver for discrete problems and linear relaxation, yielding improved convergence speed of the BB-process for several problems. [33] proposes a generic PSO-BB hybridisation for mixed discrete nonlinear programming using an SQP-based nonlinear branch and bound framework as alternating hybrids to obtain a better solution accuracy performance than both PSO and the nonlinear SQP-branch and bound framework controlled by a maximum iteration count. In the methodology, the BB-procedure would initialise an incumbent solution which PSO will use as gbest, further explore and pass on back to the BB-procedure by the time it finds a better solution and the cycle repeats. The approach yielded improved computational efficiency and solution accuracy than both PSO and the branch and bound framework, although it could not guarantee global optimality for non-convex problems [34] and thus could be outperformed by other competing methods. [35] proposes BB-PSO hybridisation for box constrained NLPs using a Lipschitzian problem-specific lower bounding scheme where a branch and bound framework would lead PSO to global optimality. Although the framework provided similar settings as the current work, the framework did not focus on computational efficiency and did not elaborate a generic approach to convex relaxation that would not require knowledge of the problem space nor offered tight bounds. The study focused on leading PSO to global optimality, however, with no emphasis on the computational efficiency of the mix and using a much looser convex relaxation approach than the scheme used in this study. [36], [37] propose hybridisation of conformational simulated annealing (CSA) with αBB as alternating hybrids to solve a highly multimodal protein structure prediction problem, a box-constrained optimisation problem, resulting in a substantial reduction in time. In a similar philosophy, as [33], the result of αBB iteration upper bound solver would seed CSA in its evolutionary strategy to aggregate much better candidate solutions upon which a next αBB iteration will take place, and the cycle repeats. However, the computational cost of conformational simulated annealing [38] is not appropriate for the type of hybridisation used in this study on top of the fact that a different methodology is used in the current work. Further comparative studies can be done on alternating and integrated hybrids. The current study extends from the work of [35] and proceeds with the hybridisation of αBB, a problem-agnostic generic branch and bound framework for twice differentiable problems with guaranteed $\epsilon-$global optimal verification where PSO is used as a substitute for the upper bound solver of classical αBB inclusive of additional optimisation in a bid to improve the computation time performance of the αBB as a unit as well as guarantee global convergence of PSO. Also, the current study relies on interval analysis as an improvement over α-convex relaxation in order to support pruning of sub-optimal regions and yields an early stop of αBB. Finally, to alleviate the local convergence shortcoming of PSO that may fail to converge to an optimum even in its vicinity [39], the final gbest at the end of each PSO search is used as an initial value to a local solver (i.e. SQP) taking advantage of gradient information. Figure 3 describes the flowchart of the hybrid algorithm.

V. COMPUTATIONAL EXPERIMENTS

To compare the performance of both αBB configurations, C^2 class functions from the CEC 2017/2019 test sets and n-dimensional classical functions from an extensive literature survey were used [40]–[42] (See Table 1). The computational

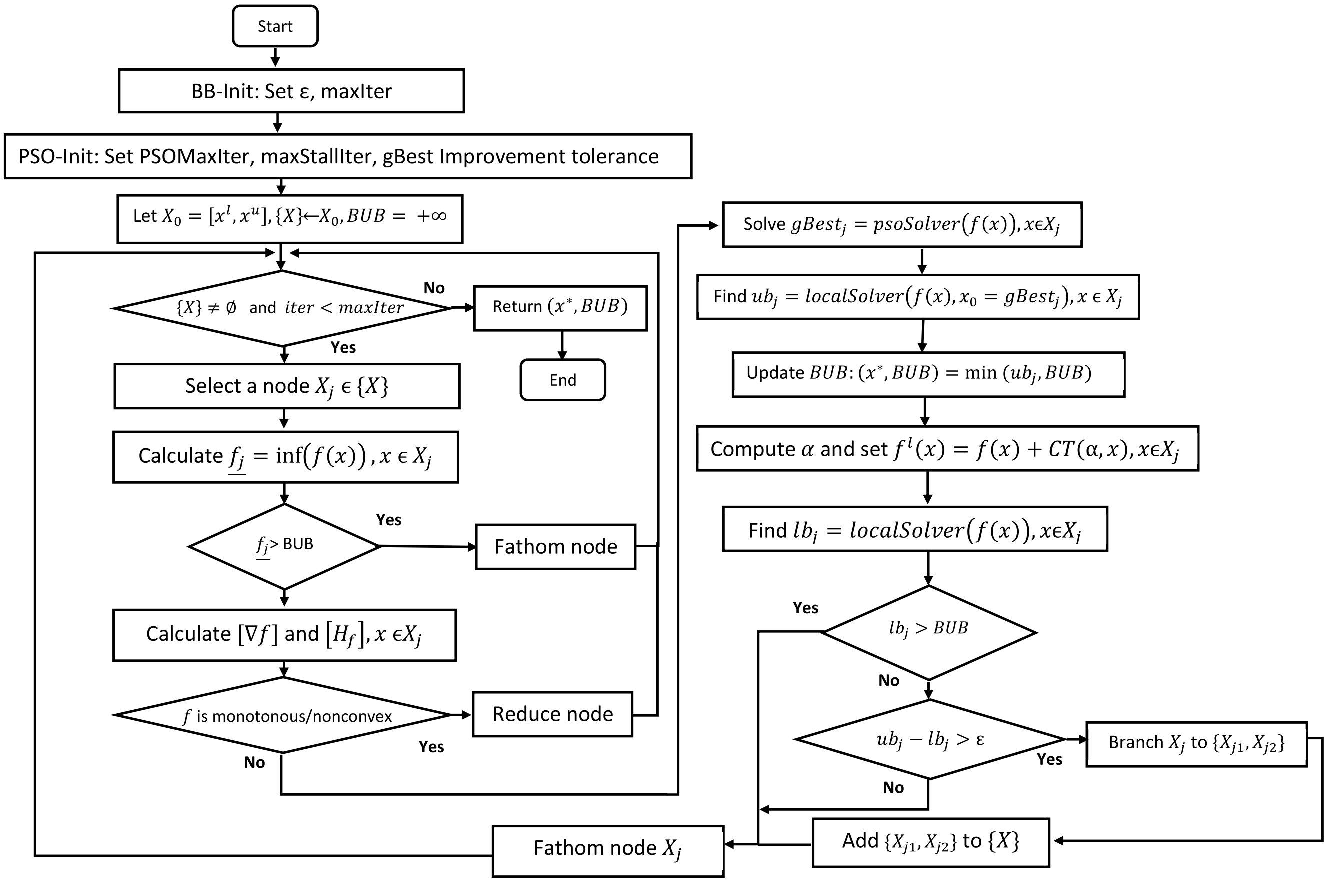

Start
BB-Init: Set ε, maxIter
PSO-Init: Set PSOMaxIter, maxStallIter, gBest Improvement tolerance
Let $X_0 = [x^l, x^u], \{X\} \leftarrow X_0, BUB = +\infty$
$\{X\} \neq \emptyset$ and $iter < maxIter$
No
Yes
Return (x^*, BUB)
End
Select a node $X_j \in \{X\}$
Calculate $\underline{f_j} = \inf(f(x)), x \in X_j$
$\underline{f_j}$> BUB
Yes
Fathom node
Calculate $[\nabla f]$ and $[H_f], x \in X_j$
f is monotonous/nonconvex
Yes
No
Reduce node
Fathom node X_j
Solve $gBest_j = psoSolver(f(x)), x \in X_j$
Find $ub_j = localSolver(f(x), x_0 = gBest_j), x \in X_j$
Update BUB: $(x^*, BUB) = \min(ub_j, BUB)$
Compute α and set $f^l(x) = f(x) + CT(\alpha, x), x \in X_j$
Find $lb_j = localSolver(f(x)), x \in X_j$
Yes
$lb_j > BUB$
No
$ub_j - lb_j > \varepsilon$
Yes
No
Branch X_j to $\{X_{j1}, X_{j2}\}$
Add $\{X_{j1}, X_{j2}\}$ to $\{X\}$

FIGURE 3. **Flowchart of the hybrid algorithm.**

experiments aimed to assess the convergence speed of αBB against the hybrid algorithm (αBB-PSO) when used as a complete search algorithm (i.e. no maximum iteration count) whereby test functions were set to dimension 2 and 3, respectively. Also, the solution accuracy of the hybrid algorithm when used as a heuristic search (i.e. maximum iteration count) was compared against PSO for the same number of function evaluations using test functions at dimension 10. Table 2 presents a summary of the configuration of the computational study. αBB was configured with a best-first search strategy using node lower bounds, a global optimum tolerance level of $\epsilon = 10^{-3}$. All pruning rules mentioned in section II-B were used. Branching was performed on the dimension with the longest size at every node. In the complete search configuration, PSO was set with 30 particles, a maximum iteration limit of 50 for dimension 2 and 100 for dimension 3. A maximum stall iteration count of 20 with a function improvement tolerance of $10^{-}3$ was set for the heuristic stop. In the heuristic search configuration, PSO was set with a maximum iteration count of 10000 with no heuristic stop compared against the hybrid algorithm with a maximum iteration count of 50 and a PSO maximum iteration count of 200 to yield the same total number of function evaluations (i.e. 10000 per particle). An adaptive inertial weight for the PSO procedure was used in all configurations according to equation 16. The cognitive and social parameters were set to $c_1 = c_2 = 2$ in line with the original publication and typical implementations [27]. The experiments were performed on a general-purpose computer: i3-core processor 64-bit @2.0GHz 8GB RAM. Interval analysis was performed using a third-party MATLAB package INTLAB (version 11) designed by [43]. The MATLAB (version 9.6.0) SQP-based NLP solver was used in this study for all local searches. In the complete search configuration, performance profiles [44] measured in terms of median CPU time were used to compare the convergence speed of the hybrid algorithm against classical αBB to assess their computational efficiency first in reaching the global optimum and second in completing the search across all test functions (See Figure 4 and 5). These results were supplemented by median CPU time performances of each function presented in Table 3 and 4. In addition, the average iteration cost of the BB-procedure in each configuration was recorded to assess the computational cost distribution of each sub-component with the branch and bound frameworks. In the heuristic configuration, the solution accuracy of PSO was compared with that of the hybrid algorithm for the same number of function evaluations using the performance profile in terms of a solution accuracy

TABLE 1. C^2 multimodal functions from CEC2017/2019 and from an extensive survey on global optimisation benchmark functions [40]–[42].

F_n	Functions	range
	CEC 2019	
F_4	Shifted and Rotated Rastrigin's Function	$[-100,100]^D$
F_5	Shifted and Rotated Griewank's Function	$[-100,100]^D$
F_6	Shifted and Rotated Weierstrass Function	$[-0.5,0.5]^D$
F_8	Shifted and Rotated Expanded Schaffer's F6	$[-100,100]^D$
F_9	Shifted and Rotated Griewank's plus Rosenbrock	$[-100,100]^D$
F_{10}	Shifted and Rotated Ackley Function	$[-100,100]^D$
	CEC 2017	
F_3	Shifted and Rotated Rosenbrock's Function	$[-100,100]^D$
F_6^*	Shifted and Rotated Schaffer's F7 Function	$[-100,100]^D$
F_{9^*}	Shifted and Rotated Levy Function	$[-100,100]^D$
HF$_1$	Hybrid Function 1	$[0,100]^D$
HF$_4$	Hybrid Function 4	$[0.1,100]^D$
HF$_9$	Hybrid Function 9	$[-0.5,0.5]^D$
	Classical n-dimensional	
SF$_4$	Modified Ackley function	$[-35,35]^D$
SF$_7$	Alpine 2 Function	$[0.1,100]^D$
SF$_{38}$	Cosine Mixture Function	$[-10,10]^D$
SF$_{43}$	Deb 1 Function	$[-100,100]^D$
SF$_{44}$	Deb 3 Function	$[0.1,100]^D$
SF$_{87}$	Pathological Function	$[-5,5]^D$
SF$_{89}$	Pinter Function	$[-100,100]^D$
SF$_{110}$	Salomon Function	$[-100,100]^D$
SF$_{133}$	Shubert Function	$[0,5]^D$
SF$_{134}$	Shubert 3 Function	$[0,5]^D$
SF$_{135}$	Shubert 4 Function	$[0,5]^D$
SF$_{144}$	Styblinski-Tang Function	$[-5,5]^D$
SF$_{153}$	Trigonometric 1 Function	$[-100,100]^D$
SF$_{154}$	Trigonometric 2 Function	$[-500,500]^D$
SF$_{165}$	W / Wavy Function	$[-100,100]^D$
SF$_{167}$	Whitley Function	$[-10,10]^D$
SF$_{171}$	Xin-She Yang Function 3	$[-20,20]^D$

metric [45] and using median final fitness values of each solver across all functions (Table 5).

Finally, bound tightness performances of both interval analysis and α-convex relaxation were recorded to assess the contribution of each lower bounding scheme in quickening convergence (See Table 6). Median scores were admitted unequal only if they were statistically significant as per the Mann-Whitney U statistical test. A significance level of 0.05 was used for the hypothesis tests ($H_0 : \mu_1 = \mu_2$). All performance metrics were obtained from an average of fifty optimisation runs.

A. DISCUSSION AND RECOMMENDATIONS

1) ON THE CONVERGENCE SPEED TO GLOBAL OPTIMUM REACH

The performance profile in Figure 4 compares the convergence speed of each solver in reaching the global optimum. The results in the figure show that the hybrid algorithm has a much better efficiency in finding the global optimum compared to the classical algorithm with a win probability ($\tau = 1$) of 85% and convergence improvement up to 36 fold. The results on Table 3 substantiate this claim as most test functions reach the global optimum earlier in the hybrid algorithm taking fewer branching iterations. These results are consistent with the rationale of the study of early global minimum reach through PSO usage and concurs with the theoretical rationales that support the superiority of meta-heuristic searches over local search techniques in terms of search capabilities.

TABLE 2. Summary of experiment configuration.

Complete search	
αBB	αBB-PSO
maxIter = $+\infty$ $\epsilon = 10^{-3}$	maxIter = $+\infty$ $\epsilon = 10^{-3}$ PSO number of particles: 30 PSO maxIter: 50 (dim 2)/ 100 (dim 3) maxStallIter: 20 gbest improvement tol: $10^{-}3$
Heurestic search	
PSO	αBB-PSO
PSO maxIter: 10000 PSO number of particles: 30 No heuristic stop SQP post-optimisation	maxIter: 200 $\epsilon = 10^{-3}$ PSO number of particles: 30 PSO maxIter: 50 (dim 10) No heuristic stop

2) ON THE OVERALL CONVERGENCE SPEED

Figure 5 presents the performance profile that compares the overall convergence speed of both αBB configurations. The results in the figure show that the hybrid algorithm has an overall better efficiency compared to the classical algorithm with a win probability of 60% and an overall convergence speed improvement of up to 38 fold. In a practical sense, the hybrid algorithm can quicken global convergence on a case to case basis up to several orders of magnitude compared to the conventional algorithm. This improvement in convergence speed is correlated with a reduction in the number of branching iterations consistent with the rationale of the study (See Table 4). The results in Table 4 shows that the hybrid algorithm reduces the number of branching iterations for 46% of test problems with statistical significance and thus is a more robust solver.

On the other hand, the hybrid algorithm did not reduce the overall execution time for several other profiles, yielding some increase in computational time. Additional calibration work should be performed to optimise the computational cost PSO as an upper bound solver to yield zero to very negligible overhead thus when no reduction in the number of branching iteration occurs. This calibration could further improve the performance profile of the hybrid algorithm. Finally, a gap can be observed between the performance profile of the hybrid algorithm to global optimum reach compared to its true overall performance profile. More investigations should be conducted on tighter lower bounding schemes that could further unleash the potential of the hybrid algorithm (Figure 4).

3) ON THE COMPUTATIONAL COST DISTRIBUTION OF BB-COMPONENTS

The results in Figure 6 presents the average computational distribution of each BB-component across both configurations recorded for test problems at dimension 2 and

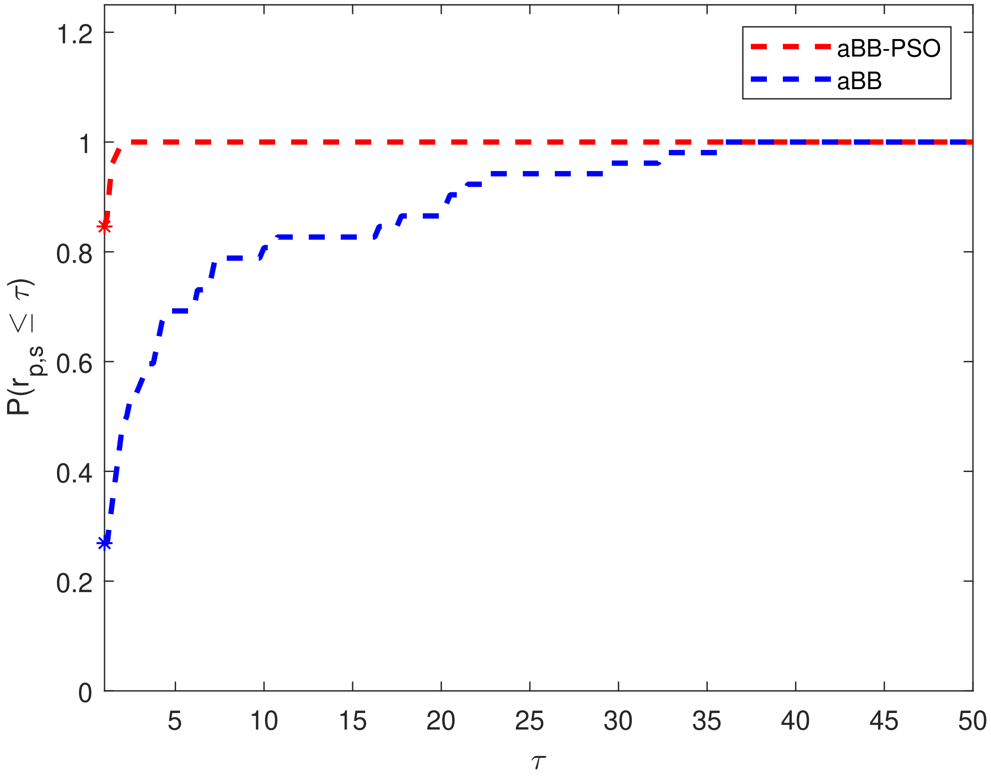


FIGURE 4. Performance profile of αBB-PSO vs αBB up to global optimum reach.

TABLE 3. Relative time to reach the global optimum.

	αBB-PSO	αBB	αBB-PSO	αBB	$\mu_{t1} \neq \mu_{t2}$
	CPU time		Iterations		p< 0.05
		Dimension 2			
F4	**0.2385±0.261**	1.0105±0.184	**4±4**	20±3	yes
F5	**0.5325±0.36**	0.7515±0.296	**8±5**	13±5	yes
F6	**24.9095±8.547**	49.13±7.455	**47±17**	105±16	yes
F8	**0.1085±0.154**	2.1755±0.531	**1±1**	20±5	yes
F9	**0.9525±0.779**	3.0205±0.682	**9±7**	30±7	yes
F10	**0.063±0.035**	1.288±0.371	**1±1**	22±6	yes
F3$_*$	**0.0505±0.034**	0.1±0.038	**1±0**	2±1	yes
F6$_*$	0.062±0.013	**0.054±0.012**	1±0	1±0	yes
F9*	**0.08±0.02**	0.301±0.182	**1±0**	4±2	yes
SF$_4$	**0.053±0.016**	1.204±0.529	**1±0**	22±10	yes
SF$_7$	**0.069±0.067**	0.417±0.118	**1±1**	8±2	yes
SF$_{38}$	0.047±0.012	**0.032±0.008**	1±0	1±0	yes
SF$_{43}$	0.0615±0.031	**0.047±0.018**	1±0	1±0	yes
SF$_{44}$	0.069±0.015	**0.054±0.034**	1±0	1±0	yes
SF$_{87}$	**0.069±0.013**	0.074±0.133	1±0	1±2	no
SF$_{89}$	**0.131±0.073**	2.3235±1.039	**1±0**	18±8	yes
SF$_{110}$	**0.107±0.029**	0.451±0.356	**2±0**	10±7	yes
SF$_{133}$	**0.1±0.105**	0.6015±1.425	**1±1**	6±13	yes
SF$_{134}$	**0.094±0.245**	0.6555±5.229	**1±2**	7±45	yes
SF$_{135}$	**0.1±0.153**	0.296±0.354	**1±1**	3±4	yes
SF$_{144}$	**0.054±0.014**	0.1±0.047	**1±0**	2±1	yes
SF$_{153}$	0.069±0.036	**0.0535±0.036**	1±0	1±1	yes
SF$_{154}$	**1.0325±0.339**	1.806±0.334	**10±3**	21±4	yes
SF$_{166}$	**0.116±0.04**	2.4815±0.73	**2±1**	41±11	yes
SF$_{167}$	**0.332±0.196**	3.31±3.372	**2±1**	20±19	yes
SF$_{171}$	**1.045±0.415**	1.5075±0.632	**12±4**	17±7	yes

	αBB-PSO	αBB	αBB-PSO	αBB	$\mu_{t1} \neq \mu_{t2}$
	CPU time		Iterations		p< 0.05
		Dimension 3			
F4	**6.9095±1.959**	7.014±1.283	**77±22**	96±18	no
F5	2.6795±0.563	**2.023±0.729**	30±6	**27±9**	yes
F6	**161.656±24.528**	187.107±27.046	**216±33**	272±39	yes
F8	**1.2885±0.95**	5.584±1.162	**8±6**	36±7	yes
F9	**13.819±3.581**	16.91±2.949	**88±23**	120±21	yes
F10	**0.172±0.095**	5.587±1.636	**2±1**	75±20	yes
F3$_*$	**0.078±0.046**	0.141±0.061	**1±0**	2±1	no
F6$_*$	0.109±0.025	**0.094±0.028**	**1±0**	1±0	no
F9$_*$	**0.219±0.466**	1.554±0.458	**2±4**	14±4	yes
SF$_4$	**0.171±0.138**	5.017±2.122	**2±1**	58±24	yes
SF$_7$	**0.906±0.414**	1.031±0.259	**10±5**	14±3	no
SF$_{38}$	**0.047±0.033**	0.109±0.064	**1±0**	2±1	yes
SF$_{43}$	0.109±0.027	**0.063±0.039**	**1±0**	1±0	yes
SF$_{44}$	0.0935±0.035	**0.078±0.045**	**1±0**	1±0	yes
SF$_{87}$	**0.2345±0.356**	0.484±1.009	**2±3**	4±8	yes
SF$_{89}$	**0.188±0.054**	1.359±0.446	**1±0**	6±2	yes
SF$_{110}$	**0.25±0.068**	0.992±0.656	**4±1**	19±12	yes
SF$_{133}$	**0.2185±0.382**	2.3275±3.039	**1±2**	15±19	yes
SF$_{134}$	**0.125±0.535**	2.036±82.303	**1±4**	15±372	yes
SF$_{135}$	**0.422±0.28**	0.9605±1.3	**3±2**	7±9	yes
SF$_{144}$	**0.078±0.025**	0.141±0.088	**1±0**	2±1	yes
SF$_{153}$	**0.125±0.148**	0.172±0.279	**1±1**	2±3	no
SF$_{154}$	**2.515±0.483**	3.853±0.658	**19±3**	31±5	yes
SF$_{166}$	**0.312±0.111**	11.1945±2.196	**4±1**	139±26	yes
SF$_{167}$	**3.207±1.314**	11.0855±30.582	**9±4**	32±85	yes
SF$_{171}$	**5.09±1.471**	7.7635±1.95	**43±12**	65±17	yes

dimension 3. It can first be observed that α-convex relaxation takes more than 60% of the computation time of a BB-iteration across all configurations. α-convex relaxation involves the computation of the interval hessian matrix $[H_f]$ as well as the estimation of the α vector. Compared to interval analysis (IA), α-convex relaxation is more costly. Moreover, regarding upper bound solvers, it can be observed that PSO (i.e. PSO + SQP fine-tune) is more computationally

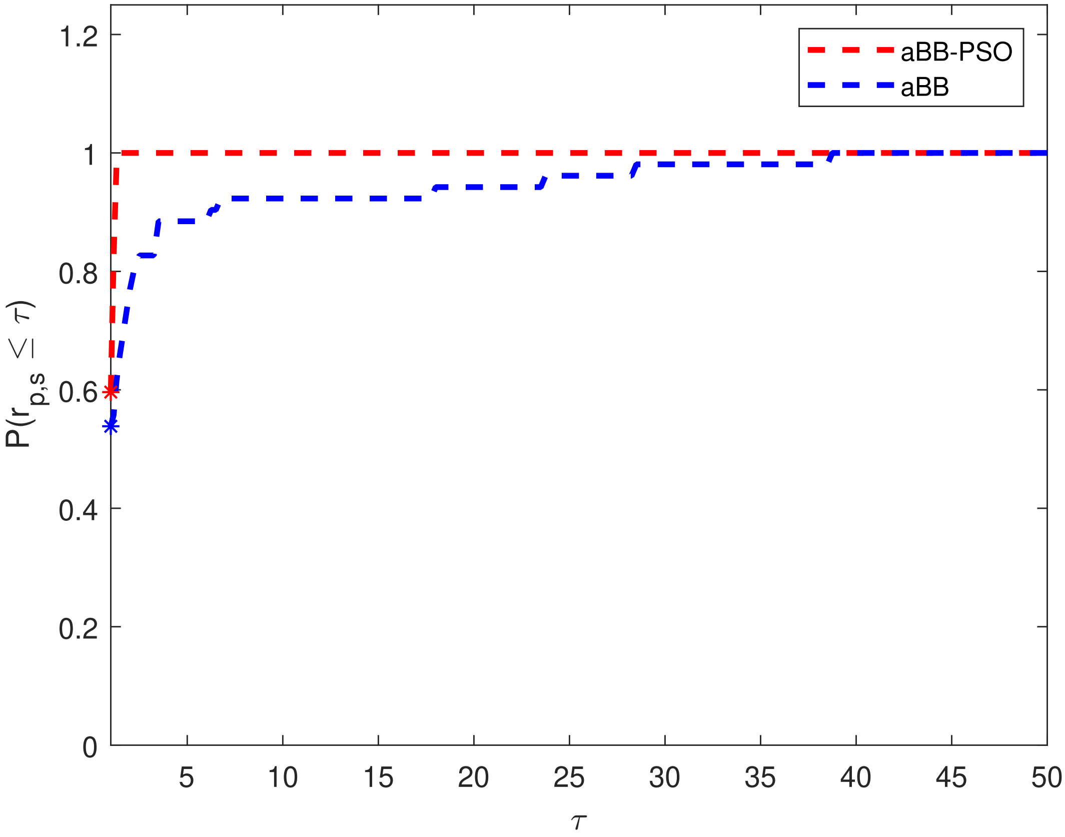


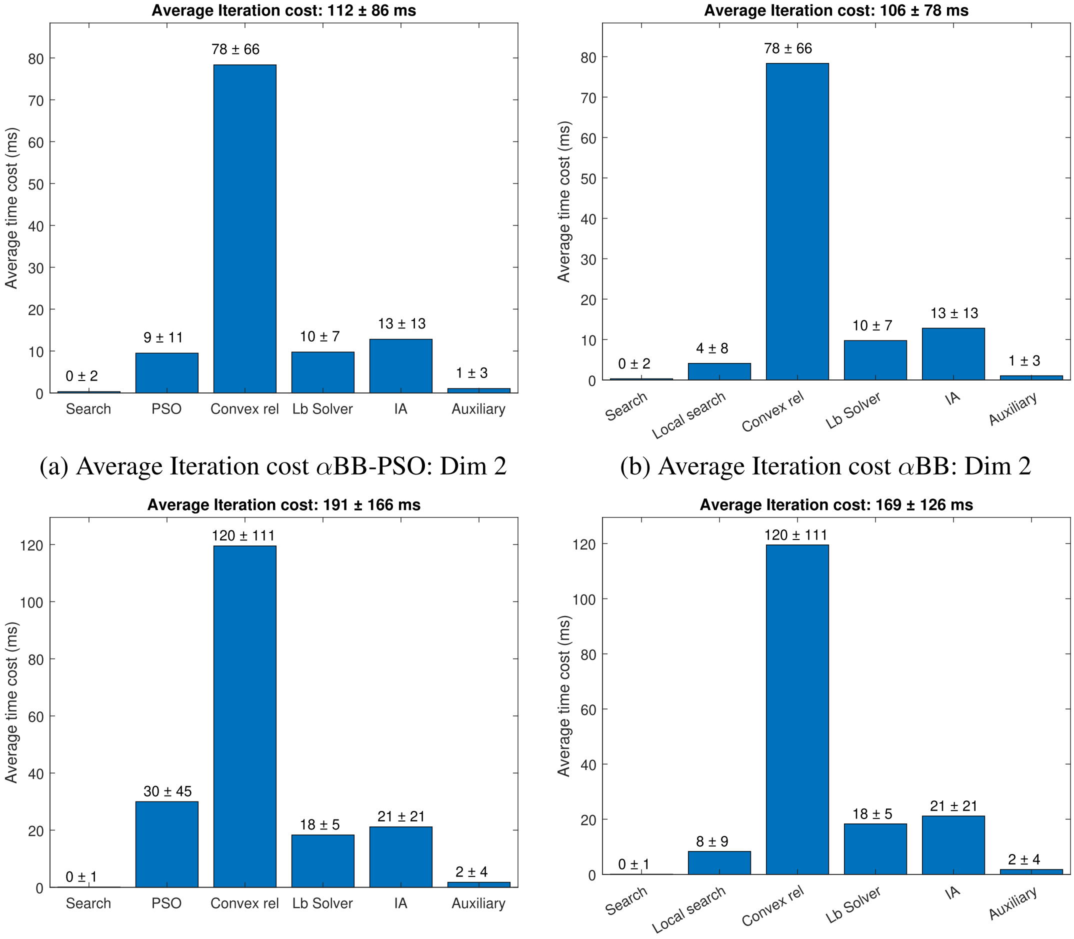
FIGURE 5. Performance profile of αBB-PSO vs αBB for the overall execution time.

TABLE 4. Overall relative time to Global optimum verification.

	αBB-PSO	αBB	αBB-PSO	αBB	$\mu_{t1} \neq \mu_{t2}$
	CPU time		Iterations		p$<$ 0.05
			Dimension 2		
F4	**0.301±0.264**	1.048±0.203	**5±4**	20±3	yes
F5	**0.7175±0.439**	0.8365±0.372	**11±5**	14±5	yes
F6	**64.8285±14.526**	111.8655±20.919	**126±17**	223±16	yes
F8	**0.124±0.277**	2.209±0.516	**1±1**	20±5	yes
F9	**1.2675±0.824**	4.125±1.379	**12±7**	41±7	yes
F10	**0.072±0.439**	2.037±0.409	**1±1**	31±6	yes
F3$_{*}$	**0.063±0.035**	0.1025±0.146	**1±0**	2±1	yes
F6$_{*}$	0.069±0.013	**0.063±0.012**	1±0	1±0	yes
F9$_{*}$	**0.0915±0.023**	0.317±0.183	**1±0**	4±2	yes
SF$_{4}$	3.895±0.594	**3.2245±0.477**	54±0	54±10	yes
SF$_{7}$	1.282±0.203	1.0185±0.153	18±1	18±2	yes
SF$_{38}$	0.9685±0.153	**0.84±0.122**	12±0	12±0	yes
SF$_{43}$	0.069±0.06	**0.053±0.032**	1±0	1±0	yes
SF$_{44}$	0.084±0.017	**0.069±0.057**	1±0	1±0	yes
SF$_{87}$	**0.084±0.014**	0.085±10.746	1±0	1±2	no
SF$_{89}$	**0.139±0.074**	3.2735±1.195	**1±0**	25±8	yes
SF$_{110}$	**0.69±0.37**	1.2335±0.397	**12±0**	23±7	yes
SF$_{133}$	46.7165±6.607	**43.9565±6.369**	309±1	309±13	yes
SF$_{134}$	21.7045±3.127	**19.4635±2.951**	160±2	160±45	yes
SF$_{135}$	8.9275±1.317	**8.3775±1.228**	77±1	77±4	yes
SF$_{144}$	1.921±0.3	**1.705±0.254**	28±0	28±1	yes
SF$_{153}$	0.069±1.809	**0.069±1.908**	1±0	1±1	no
SF$_{154}$	**1.049±0.341**	1.842±0.34	**10±3**	21±4	yes
SF$_{166}$	**1.547±0.761**	3.396±0.662	**25±1**	53±11	yes
SF$_{167}$	72.9505±10.368	**69.606±9.873**	338±1	338±19	yes
SF$_{171}$	**2.1035±0.374**	2.596±0.454	**20±4**	27±7	yes

	αBB-PSO	αBB	αBB-PSO	αBB	$\mu_{t1} \neq \mu_{t2}$
	CPU time		Iterations		p$<$ 0.05
			Dimension 3		
F4	**8.3185±2.094**	8.5695±1.445	**93±22**	110±18	no
F5	3.2805±0.69	**2.702±1.095**	37±6	**35±9**	yes
F6	**281.9035±66.37**	564.695±133.334	**378±33**	748±39	yes
F8	**2.7625±1.03**	6.3545±1.272	**17±6**	39±7	yes
F9	**17.9965±3.878**	21.1425±4.65	**113±23**	140±21	yes
F10	**4.6865±2.059**	9.026±1.473	**57±1**	104±20	yes
F3$_{*}$	**0.094±0.049**	0.1565±0.062	**1±0**	2±1	no
F6$_{*}$	0.11±0.038	**0.109±0.047**	1±0	1±0	no
F9$_{*}$	**0.235±0.47**	1.5855±0.466	**2±4**	14±4	yes
SF$_{4}$	60.2105±8.565	**56.812±8.049**	428±1	428±24	yes
SF$_{7}$	1.984±0.294	**1.656±0.242**	22±5	22±3	yes
SF$_{38}$	1.031±0.155	**0.89±0.136**	13±0	13±1	yes
SF$_{43}$	0.078±0.014	**0.078±0.166**	1±0	1±0	no
SF$_{44}$	0.109±0.126	**0.094±0.093**	1±0	1±0	yes
SF$_{87}$	**0.3825±32.228**	14.741±71.211	**2±3**	113±8	yes
SF$_{89}$	**1.3985±3.116**	8.636±2.956	**7±0**	44±0	yes
SF$_{110}$	**2.817±1.087**	3.8975±1.064	**44±1**	61±12	yes
SF$_{133}$	128.2665±18.236	**121.2255±17.192**	577±0	577±0	yes
SF$_{134}$	462.572±65.487	**438.4995±62.163**	1499±4	1499±372	yes
SF$_{135}$	89.529±12.677	**86.635±12.319**	459±2	459±9	yes
SF$_{144}$	11.3285±1.63	**10.3145±1.512**	120±0	120±1	yes
SF$_{153}$	**0.484±187.728**	365.84±186.189	**3±0**	2001±0	no
SF$_{154}$	**2.5535±0.489**	3.921±0.677	**19±3**	31±5	yes
SF$_{166}$	**11.32±3.711**	16.038±2.442	**122±1**	170±26	yes
SF$_{167}$	498.1875±70.883	**486.27±68.934**	1083±4	1083±85	yes
SF$_{171}$	**9.783±1.66**	11.122±1.681	**74±12**	85±17	yes

expensive than the classical SQP-based local search of the αBB configuration yielding additional computational overhead responsible for its improved search capabilities; however, also susceptible to deteriorate computational



(a) Average Iteration cost αBB-PSO: Dim 2

(b) Average Iteration cost αBB: Dim 2

(c) Average Iteration cost αBB-PSO:Dim 3

(d) Average Iteration cost αBB:Dim 3

FIGURE 6. Average distribution cost in αBB and αBB-PSO.

efficiency. As mentioned in section V-A2, this computational overhead can be further minimised or eliminated by calibration of PSO parameters (i.e. heuristic stop parameters, number of particles, maximum iteration count) or potentially by looking at the computational contribution of the SQP fine-tuning mechanism. The lower bound optimisation (Lb Solver) has been the third-largest contributor in the BB-iteration cost. The computational cost of the node selection process and auxiliary algorithmic routines were very negligible across all configurations.

4) ON THE SOLUTION ACCURACY OF THE αBB-PSO HEURISTIC CONFIGURATION

The performance profile in Figure 7 and results in Table 5 assess the capabilities of the hybrid algorithm as a heuristic search (i.e. maximum iteration limit) against PSO in terms of solution accuracy. This experiment was performed on equal grounds guaranteeing that both PSO and the hybrid will perform an equal number of function evaluations (i.e. 10000 per particle), and the results of both solvers were fine-tuned by an SQP solver subsequently. It can be observed that the hybrid algorithm outperforms PSO in terms of solution accuracy yielding a performance profile with a win probability of 100% and an improvement in solution accuracy in excess of 100 fold. In addition, the hybrid algorithm could guarantee complete search in several instances (F_3, F_6, SF_{38}, SF_{43}, SF_{44} and SF_{154}), thus ending the search with confidence of exhaustive exploration. The above results exemplify the benefits of combining the search capability of the branch and bound framework with that of PSO in the same vein as [35]. On the other hand, PSO presented a better time efficiency than the hybrid algorithm. To further improve the computational efficiency of the hybrid algorithm, interval analysis could be used as the sole lower bounding scheme because statistically, it yields much tighter and more computationally efficient than α-convex relaxation (Figure 6 and Table 6).

5) ON THE CONTRIBUTION OF INTERVAL ANALYSIS TO REGION PRUNING

The results in Table 6 show a comparison of bound tightness between interval analysis and the $\alpha-$convex relaxation over

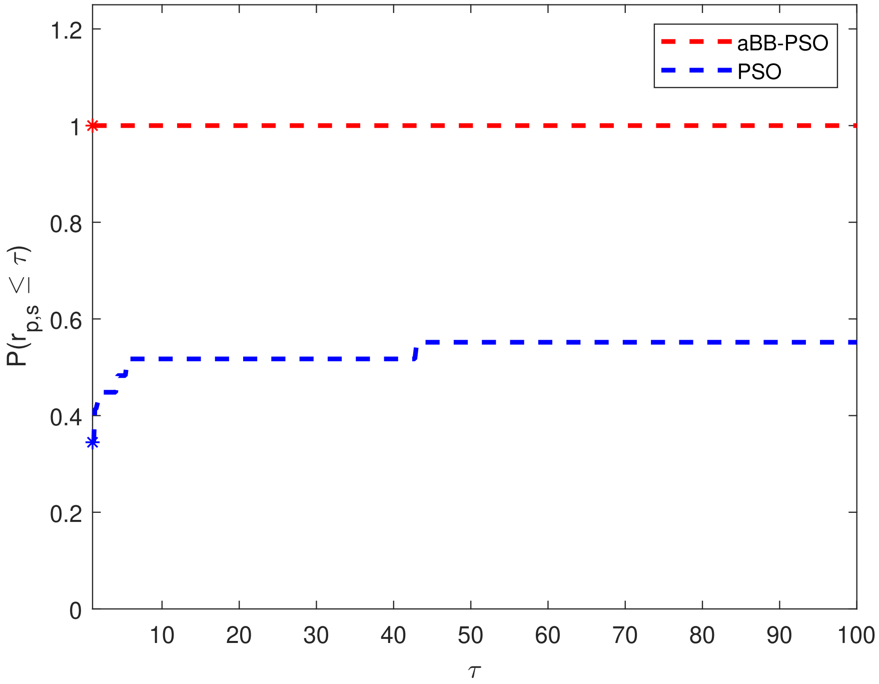


FIGURE 7. Performance profile of PSO against heuristic αBB-PSO in terms of solution accuracy.

TABLE 5. Solution accuracy: Heurestic αBB-PSO vs PSO. Max Function eval: 10000.

	αBB-PSO	PSO	αBB-PSO	PSO	αBB-PSO	$\mu_1 \neq \mu_2$
	Global optimum		CPU time (s)		Iteration	$p < 0.05$
			Dimension 10			
F4	**49.022±22.152**	262.165±266.646	57.8745±0.647	**4.5065±0.208**	200±0	yes
F5	**1.127±0.066**	1.531±0.326	62.3865±0.431	**4.6115±0.311**	200±0	yes
F6	**3.615±0.442**	5.219±2.066	587.6705±3.071	**45.295±0.335**	200±0	yes
F8	**3.156±0.200**	3.567±0.505	125.0565±1.139	**5.17±0.249**	200±0	yes
F9	**211.159±224.653**	9.0e+03±2.3e+04	103.949±1.376	**5.7645±0.291**	200±0	yes
F10	**17.491±7.574**	21±0.000	45.142±0.56	**4.108±0.209**	200±0	yes
$F3_*$	**1±0.000**	2.484±1.778	**1.3805±0.609**	3.889±0.245	**4±2***	yes
$F6_*$	1±0.000	1±0.000	**0.547±1.09**	4.8945±0.193	**1±1***	no
$F9_*$	**1±0.000**	19.534±131.056	27.007±3.993	**6.9695±0.335**	64±9	no
HF_1	**20.236±12.095**	215.871±186.330	90.6285±1.003	**17.092±0.651**	200±0	yes
HF_4	**23.524±4.761**	297.093±427.033	80.522±1.479	**20.243±0.712**	200±0	yes
HF_9	**1.145±0.051**	3.728±3.099	207.556±3.49	**34.736±0.932**	200±0	yes
SF_4	-22.259±0.331	**-22.318±0.250**	79.616±2.044	**4.5685±0.17**	200±0	no
SF_7	**-9.3e+09±0.003**	-9.1e+09±3.1e+08	77.276±0.707	**4.5135±0.247**	200±0	no
SF_{38}	-1.0e+03±0.000	-1.0e+03±0.000	**0.188±0.014**	4.8945±0.245	**1±0***	no
SF_{43}	**-1±0.000**	-0.996±0.020	**0.299±1.402**	5.193±0.187	**1±3***	no
SF_{44}	**-1±0.000**	-0.924±0.048	**4.1635±45.892**	6.0035±0.185	**6±69***	yes
SF_{87}	**-8.908±0.044**	-8.802±0.303	121.3435±0.852	**4.475±0.245**	200±0	no
SF_{89}	**0±0.000**	130.753±714.819	154.6425±1.087	**5.2645±0.258**	200±0	yes
SF_{110}	**0±0.000**	0.106±0.024	29.8845±4.326	**4.3085±0.257**	200±28	yes
SF_{133}	**-4.9e+10±1.2e+10**	-1.3e+10±1.7e+10	135.093±1.043	**4.957±0.234**	200±0	yes
SF_{134}	**-102.569±0.000**	-97.667±4.286	108.8985±0.936	**4.9335±0.199**	200±0	yes
SF_{135}	**-122.1±3.898**	-104.312±10.373	111.017±1.22	**4.9545±0.217**	200±0	yes
SF_{144}	-391.662±0.000	-391.662±0.000	61.2565±1.447	**4.992±0.264**	200±0	no
SF_{153}	**0.977±0.879**	33.454±29.774	74.152±1.842	**4.863±0.316**	200±0	yes
SF_{154}	1±0.000	1±0.000	43.776±2.544	**4.4285±0.199**	**87±5***	no
SF_{166}	**0.096±0.021**	0.162±0.103	61.8345±0.576	**4.4335±0.229**	200±0	yes
SF_{167}	**-99.303±0.000**	-91.484±8.258	844.677±6.227	**5.8375±0.298**	200±0	yes
SF_{171}	0±0.000	0±0.000	87.653±0.782	**5.326±0.193**	200±0	no

TABLE 6. Bound tightness comparison (Average of 50 optimisation runs, Dimension 3).

	αBB-PSO					αBB				
	$\underline{f_i}> lb_i$	$lb_i <$ f$_i$		lb_i-$\underline{f_i}$		$\underline{f_i}> lb_i$	$lb_i <$ f$_i$		lb_i-$\underline{f_i}$	
			min	median	max			min	median	max
F4	**89**	1	-1.5e+07	**-5.7e+04**	1.9e-01	**102**	1	-1.5e+07	**-2.8e+04**	1.9e-01
F5	**36**	0	-6.6e+04	**-590.2079**	-3.5e-02	**38**	0	-6.6e+04	**-652.0783**	-3.5e-02
F6	**390**	0	-5.5e+14	**-2.6e+08**	-10.2537	**757**	0	-5.5e+14	**-1.5e+04**	-6.2e-01
F8	**15**	0	-7.9e+14	**-2.4e+12**	-2.2e+07	**39**	0	-7.9e+14	**-2.5e+08**	-115.4784
F9	**106**	0	-3.8e+23	**-3.3e+12**	-961.117	**142**	0	-3.8e+23	**-1.0e+11**	-2.0262
F10	**56**	0	$-\infty$	$-\infty$	$-\infty$	**104**	0	$-\infty$	$-\infty$	$-\infty$
F3$_*$	**2**	0	-8.1e+10	**-8.1e+10**	-4.1e+10	**2**	0	-8.1e+10	**-6.1e+10**	-3.3e+08
F6$_*$	**2**	0	$-\infty$	$-\infty$	$-\infty$	**2**	0	$-\infty$	$-\infty$	$-\infty$
F9$_*$	**4**	0	-1.9e+08	**-1.6e+08**	-2.4e+03	**13**	0	-1.9e+08	**-2.3e+06**	-39.9679
SF$_4$	**363**	64	$-\infty$	$-\infty$	2.2862	**363**	64	$-\infty$	$-\infty$	2.2862
SF$_7$	**21**	1	-1.8e+07	**-1.1e+05**	3.835	**21**	1	-1.8e+07	**-1.1e+05**	3.835
SF$_{38}$	**12**	1	-7.7e+03	**-3.4e+03**	6.8e-13	**12**	1	-7.7e+03	**-3.4e+03**	6.8e-13
SF$_{43}$	**1**	0	-4.4e+07	**-4.4e+07**	-4.4e+07	**2**	0	-4.4e+07	**-4.4e+07**	-2.1e+06
SF$_{44}$	**2**	0	-6.2e+11	**-6.2e+11**	-3.7e+10	**2**	0	-6.2e+11	**-6.2e+11**	-2.1e+11
SF$_{87}$	**108**	3	$-\infty$	$-\infty$	5.8e-03	**326**	10	$-\infty$	$-\infty$	8.0e-03
SF$_{89}$	**18**	0	-3.1e+19	**-7.0e+18**	-8.4e+08	**45**	0	-3.1e+19	**-2.3e+16**	-4.6057
SF$_{110}$	**42**	0	$-\infty$	$-\infty$	$-\infty$	**62**	0	$-\infty$	$-\infty$	$-\infty$
SF$_{133}$	**570**	6	-2.1e+06	**-5.0e+04**	273.7271	**570**	6	-2.1e+06	**-5.0e+04**	273.7271
SF$_{134}$	**1.3e+03**	159	-3.2e+03	**-10.8472**	9.8503	**1.3e+03**	160	-3.2e+03	**-10.8472**	9.8503
SF$_{135}$	**448**	10	-3.2e+03	**-104.3865**	9.1891	**448**	10	-3.2e+03	**-104.3865**	9.1891
SF$_{144}$	7	**113**	-5.6e+03	296.3551	447.5679	7	**113**	-5.6e+03	296.3551	447.5679
SF$_{153}$	**881**	0	-4.2e+06	**-3.6e+06**	-1.5e-01	**1.1e+03**	0	-4.2e+06	**-1.5e+04**	-686.1572
SF$_{154}$	**19**	0	-2.4e+15	**-2.3e+12**	-2.7e+07	**31**	0	-2.4e+15	**-8.5e+09**	-205.1114
SF$_{166}$	**108**	0	-6.1e+07	**-9.9e+03**	-5.8e-02	**171**	0	-6.1e+07	**-9.9e+03**	-5.8e-02
SF$_{167}$	**1.0e+03**	63	-9.6e+13	**-317.683**	1.5355	**1.0e+03**	63	-9.6e+13	**-317.683**	1.5355
SF$_{171}$	**73**	0	-6.5e+06	**-1.1e+05**	-61.4916	**85**	0	-6.5e+06	**-6.6e+04**	-61.4916

a series of fifty optimisation runs, counting the number of times interval analysis had attained a tighter lower bound than α-convex relaxation and vice-versa. It also evaluated the gap between the two underestimators in each iteration during BB-procedures. The results show how interval analysis was much tighter than α-convex relaxation on most lower bounding routines for most test profiles (except SF$_{144}$). In most test profiles, interval analysis totally outperformed convex relaxation in finding tight bounds. Also, in instances where α-convex relaxation was unable to find a finite lower bound (i.e. $lb_i = -\infty$), interval analysis supplemented the shortcoming. Hence, interval analysis was able to anticipate tighter bounds much earlier than α-convex relaxation, which eventually led to early pruning to quicken the overall convergence of the hybrid algorithm.

However, it should also be noted that interval analysis does not always lead to tight bounds as in many instances, it also leads to overestimation as caused by the dependency problem related to the inner structure of the problem algebraic expression [20]. This can also explain why the hybrid algorithm did not manage to quicken convergence for several other test profiles, albeit reaching the global optimum earlier in most cases (See Table 3). Further study should be performed towards the improvement of classical interval analysis or the implementation of other interval analysis approaches reported in the literature such as Taylor model-based interval analysis [46], [47] or affine interval analysis [48]. Such an improvement in accuracy will not only improve lower bounding by interval analysis, but it will also ameliorate the tightness of α-convex relaxation, which typically depends on interval analysis (i.e. Interval Hessian matrix). In addition, possible restructuring of the problem algebraic expression [49], [50] can also contribute to improving the bound tightness of classical interval analysis and α-convex relaxation.

VI. CONCLUSION

The current study has proposed the hybridisation of particle swarm optimisation with αBB in a bid to improve the convergence speed of the branch and bound framework, on the one hand, and to obtain a better heuristic solver on the other hand. It has shown an improvement in convergence speed of the branch and bound framework on several test profiles owing to an early solution discovery by PSO and owing to the availability of sharp lower bounds supplemented by interval analysis. Also, it has shown a drastic improvement in solution accuracy when PSO is used via branch and bound framework, combining the search capabilities of both devices. Interval analysis has been very determinant in the proposed algorithm yielding much tighter bounds than α-convex relaxation and at a better computational efficiency, therefore, hinting that it could be used as its substitute to further improve computational efficiency. More investigations should be performed on tighter bounding schemes in interval analysis which would subsequently boost the convergence speed of the hybrid algorithm. Additional calibration work should be conducted to improve the computational cost of PSO as an upper bound solver in order to minimise computational overhead and ameliorate the efficiency of the hybrid algorithm.

REFERENCES

[1] A. Neumaier, "Complete search in continuous global optimization and constraint satisfaction," *Acta Numerica*, vol. 13, pp. 271–369, May 2004.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE ICNN*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.

[3] S. Mirjalili, "Genetic algorithm," in *Evolutionary Algorithms and Neural Networks*. Cham, Switzerland: Springer, 2019, pp. 43–55.

[4] K. V. Price, "Differential evolution," in *Handbook of Optimization*. Cham, Switzerland: Springer, 2013, pp. 187–214.

[5] K. A. Dowsland and J. Thompson, "Simulated annealing," in *Handbook of Natural Computing*. Berlin, Germany: Springer, 2012, pp. 1623–1655.

[6] M. Dorigo and T. Stützle, "Ant colony optimization: Overview and recent advances," in *Handbook of Metaheuristics*. Boston, MA, USA: Springer, 2019, pp. 311–351.

[7] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci.*, vol. 237, pp. 82–117, Jul. 2013.

[8] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optim.*, vol. 19, pp. 79–102, Feb. 2016.

[9] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier, "A global optimization method, αBB, for general twice-differentiable constrained NLPs—I. Theoretical advances," *Comput. Chem. Eng.*, vol. 22, no. 9, pp. 1137–1158, Aug. 1998.

[10] C. S. Adjiman, I. P. Androulakis, C. D. Maranas, and C. A. Floudas, "A global optimization method, αBB, for process design," *Comput. Chem. Eng.*, vol. 20, pp. S419–S424, Jan. 1996.

[11] C. E. Gounaris and C. A. Floudas, "Tight convex underestimators for $\mathcal{C}^2$-continuous problems: II. Multivariate functions," *J. Global Optim.*, vol. 42, no. 1, pp. 69–89, Sep. 2008.

[12] N. Kazazakis and C. S. Adjiman, "Arbitrarily tight α BB underestimators of general non-linear functions over sub-optimal domains," *J. Global Optim.*, vol. 71, no. 4, pp. 815–844, 2018.

[13] M. Hladík, "On the efficient Gerschgorin inclusion usage in the global optimization αBB method," *J. Global Optim.*, vol. 61, no. 2, pp. 235–253, 2015.

[14] M. Hladík, D. Daney, and E. Tsigaridas, "A filtering method for the interval eigenvalue problem," *Appl. Math. Comput.*, vol. 217, no. 12, pp. 5236–5242, Feb. 2011.

[15] M. Hladík, "Bounds on eigenvalues of real and complex interval matrices," *Appl. Math. Comput.*, vol. 219, no. 10, pp. 5584–5591, 2013.

[16] N. Kazazakis, "Parallel computing, interval derivative methods, heuristic algorithms, and their implementation in a numerical solver, for deterministic global optimization," Ph.D. dissertation, Dept. Chem. Eng., Imperial College London, London, U.K., 2016.

[17] C. D. Maranas and C. A. Floudas, "A deterministic global optimization approach for molecular structure determination," *J. Chem. Phys.*, vol. 100, no. 2, pp. 1247–1261, Jan. 1994.

[18] C. D. Maranas and C. A. Floudas, "Global minimum potential energy conformations of small molecules," *J. Global Optim.*, vol. 4, no. 2, pp. 135–170, Mar. 1994.

[19] M. Althoff and D. Grebenyuk, "Implementation of interval arithmetic in CORA 2016," in *Proc. EPiC Ser. Comput.*, 2016, pp. 91–105.

[20] H. Dawood, *Theories of Interval Arithmetic: Mathematical Foundations and Applications*. Germany: Lambert Acad. Publishing, 2011.

[21] Y. Puranik and N. V. Sahinidis, "Domain reduction techniques for global NLP and MINLP optimization," *Constraints*, vol. 22, no. 3, pp. 338–376, Jul. 2017.

[22] I. Mazhoud, K. Hadj-Hamou, J. Bigeon, and G. Remy, "Interval-based global optimization in engineering using model reformulation and constraint propagation," *Eng. Appl. Artif. Intell.*, vol. 25, no. 2, pp. 404–417, Mar. 2012.

[23] H. Ratschek and R. L. Voller, "What can interval analysis do for global optimization?" *J. Global Optim.*, vol. 1, no. 2, pp. 111–130, 1991.

[24] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," *Evol. Comput.*, vol. 25, no. 1, pp. 1–54, 2017.

[25] J. C. Bansal, *Particle Swarm Optimization*. Cham, Switzerland: Springer, 2019, pp. 11–23, doi: 10.1007/978-3-319-91341-4_2.

[26] S. Mirjalili, *Particle Swarm Optimisation*. Cham, Switzerland: Springer, 2019, pp. 15–31, doi: 10.1007/978-3-319-93025-1_2.

[27] A. R. Jordehi and J. Jasni, "Parameter selection in particle swarm optimisation: A survey," *J. Experim. Theor. Artif. Intell.*, vol. 25, no. 4, pp. 527–542, 2013.

[28] Y. Cao, H. Zhang, W. Li, M. Zhou, Y. Zhang, and W. A. Chaovalitwongse, "Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions," *IEEE Trans. Evolutionary Comput.*, vol. 23, no. 4, pp. 718–731, Aug. 2018.

[29] M. E. H. Pedersen, "Good parameters for particle swarm optimization," Hvass Laboratories, Copenhagen, Denmark, Tech. Rep. HL1001, 2010.

[30] K. Zielinski and R. Laur, "Stopping criteria for a constrained single-objective particle swarm optimization algorithm," *Informatica*, vol. 31, no. 1, pp. 51–59, 2007.

[31] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A," *J. ACM*, vol. 32, no. 3, pp. 505–536, 1985.

[32] Y. Gao, Z. Ren, and C. Xu, "A branch and bound-PSO hybrid algorithm for solving integer separable concave programming problems," *Appl. Math. Sci.*, vol. 1, no. 11, pp. 517–525, 2007.

[33] S. Nema, J. Goulermas, G. Sparrow, and P. Cook, "A hybrid particle swarm branch-and-bound (HPB) optimizer for mixed discrete nonlinear programming," *IEEE Trans. Syst., Man, A, Syst. Humans*, vol. 38, no. 6, pp. 1411–1424, Nov. 2008.

[34] S. Leyffer, "Integrating SQP and branch-and-bound for mixed integer nonlinear programming," *Comput. Optim. Appl.*, vol. 18, no. 3, pp. 295–309, 2001.

[35] Z. Tang and K. K. Bagchi, "Globally convergent particle swarm optimization via branch-and-bound," *Comput. Inf. Sci.*, vol. 3, no. 4, pp. 60–71, Oct. 2010.

[36] J. L. Klepeis, M. J. Pieja, and C. A. Floudas, "A new class of hybrid global optimization algorithms for peptide structure prediction: Integrated hybrids," *Comput. Phys. Commun.*, vol. 151, no. 2, pp. 121–140, Mar. 2003.

[37] J. L. Klepeis, M. J. Pieja, and C. A. Floudas, "Hybrid global optimization algorithms for protein structure prediction: Alternating hybrids," *Biophysical J.*, vol. 84, no. 2, pp. 869–882, Feb. 2003.

[38] I. Joung, J. Y. Kim, S. P. Gross, K. Joo, and J. Lee, "Conformational space annealing explained: A general optimization algorithm, with diverse applications," *Comput. Phys. Commun.*, vol. 223, pp. 28–33, Feb. 2018.

[39] F. van den Bergh and A. P. Engelbrecht, "A new locally convergent particle swarm optimiser," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, vol. 3, Oct. 2002, p. 6.

[40] K. Price, N. Awad, M. Ali, and P. Suganthan, "Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization," Nanyang Technol. Univ., Singapore, Tech. Rep., 2018.

[41] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *Int. J. Math. Modeling Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.

[42] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," Nat. Univ. Defense Technol., Kyungpook Nat. Univ., Nanyang Technol. Univ., Singapore, Tech. Rep., 2017.

[43] S. Rump, "INTLAB—INTerval LABoratory," in *Developments in Reliable Computing*, T. Csendes, Ed. Dordrecht, The Netherlands: Kluwer, 1999, pp. 77–104. [Online]. Available: http://www.ti3.tuhh.de/rump/

[44] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, no. 2, pp. 201–213, 2002.

[45] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," *J. Global Optim.*, vol. 31, no. 4, pp. 635–672, 2005.

[46] M. Berz and G. Hoffstätter, "Computation and application of Taylor polynomials with interval remainder bounds," *Reliable Comput.*, vol. 4, no. 1, pp. 83–97, 1998.

[47] M. Gavriliu, "Towards more efficient interval analysis: Corner forms and a remainder interval Newton method," Ph.D. dissertation, California Inst. Technol., Pasadena, CA, USA, 2005.

[48] D. Degrauwe, G. Lombaert, and G. De Roeck, "Improving interval analysis in finite element calculations by means of affine arithmetic," *Comput. Struct.*, vol. 88, nos. 3–4, pp. 247–254, Feb. 2010.

[49] M. Hladík, "The effect of Hessian evaluations in the global optimization αBB method," in *Modeling, Simulation and Optimization of Complex Processes HPSC 2015*. Cham, Switzerland: Springer, 2017, pp. 67–79.

[50] M. Ceberio and V. Kreinovich, "Greedy algorithms for optimizing multivariate horner schemes," *ACM SIGSAM Bull.*, vol. 38, no. 1, pp. 8–15, Mar. 2004.

YVES MATANGA received the B.Tech. degree in electrical engineering (specialization in electronics) from the Tshwane University of Technology, Pretoria, South Africa, in 2014, the M.Sc. degree in electrical and electronic systems from ESIEE, France, and the M.Tech. degree in electrical engineering from the Tshwane University of Technology, in 2018. He is currently pursuing the Ph.D. degree in electrical and electronic engineering with the University of Johannesburg, South Africa, working on convergence improvement of deterministic and globally convergent optimization algorithms with applications to control systems. He has previously published in Elsevier and the IEEE Africon Conference. His research interests include dynamic systems and control, optimization, artificial intelligence, and signal and image processing.

YANXIA SUN (Member, IEEE) received the joint D.Tech. degree in electrical engineering from the Tshwane University of Technology, South Africa, and the Ph.D. degree in computer science from University Paris-EST, France, in 2012. She is currently working as a Professor with the Department of Electrical and Electronic Engineering Science, University of Johannesburg, South Africa. She has more than 15 years of teaching and research experience. She has lectured five courses in the universities. She has supervised or co-supervised six postgraduate projects to completion. She has published 110 articles, including 35 ISI master indexed journal articles. She is the investigator or co-investigator for six research projects. Her research interests include renewable energy, evolutionary optimization, neural networks, nonlinear dynamics, and control systems. She is a member of the South African Young Academy of Science (SAYAS).

ZENGHUI WANG (Member, IEEE) received the B.Eng. degree in automation from the Naval Aviation Engineering Academy, China, in 2002, and the Ph.D. degree in control theory and control engineering from Nankai University, China, in 2007. He is currently a Professor with the Department of Electrical and Mining Engineering, University of South Africa (UNISA), South Africa. His research interests include industry 4.0, control theory and control engineering, engineering optimization, image/video processing, artificial intelligence, and chaos.